**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ**

**И РАДИОЭЛЕКТРОННИКИ**

Факультет инновационного непрерывного образования

Специальность: Программное обеспечение информационных технологий.

**Курсовой проект**

по дисциплине«Операционные системы и системное программирование. Часть 2»

тема курсового проекта: «Медицинская программа».

**Исполнитель:** студент группы 691051

Засимук Н. С.

**Руководитель:** Мельникова Елена Владимировна.

Минск 2019

**1.Содержание.**

1. Содержание 2

2.Задание на курсовой проект 3

3.Введение. 3

4. Аналитический обзор литературы 4

5. Моделирование предметной области 8

5.1 Теоретические сведения по теме 8

5.2 Обоснование выбора языка программирования 12

6. Проектирование программного средства 14

7. Тестирование ПС 21

8. Руководство пользователя 24

9. Заключение 25

10. Список использованных источников 25

11. Листинг программы 26

**2.Задание на курсовой проект.**

Разработать программу для ведения врачом на амбулаторном приеме электронной медицинской карты пациента.

**3.Введение.**

Актуальность создания информационной системы в медицинских учреждениях обусловлена сегодня необходимостью использования больших, и постоянно растущих, объемов информации при решении диагностических, терапевтических, статистических, управленческих и других задач.

Информатизация деятельности учреждений здравоохранения уже давно стала не просто данью современных веяний, а насущной необходимостью. Обработка все время растущих массивов информации стала возможна только с использованием современных компьютерных технологий.

Ни для кого не секрет, что большая часть приема уходит не на решение клинических вопросов, а на сопроводительную и далеко не самую основную работу – оформление поликлинических талонов и другой отчетной документации, записей в амбулаторной карте или истории болезни, назначений консультаций или обследования и т.д. Уже не вызывает сомнений, что наиболее эффективным инструментов для облегчения труда медицинских сотрудников и повышения его эффективности являются компьютерные технологии. Автоматизация способна не просто облегчить работу, она должна освободить персонал от рутины и дать ему принципиально новый инструмент, который прямо или косвенно, но приведет к сокращению нецелевого расхода интеллектуального багажа, реализации желания работать и заниматься именно медициной.

Основная цель курсовой работы - сократить временные затраты регистратора за счет автоматизации основных процессов:

* ведение электронной амбулаторной карты пациента
* регистрация врачебных посещений
* ведение дневника врача
* регистрация медицинских осмотров

**4.Аналитический обзор литературы**

**Анализ существующих аналогов**

Автоматизация с помощью макросов

При работе с документацией чаще всего используется Microsoft Officе. Автоматизация документов Office возможна с помощью различных макросов. Обзоры макросов будут описаны с помощью VBA.

Можно автоматизировать практически любую рутинную работу и повторяющуюся работу. Можно выполнить эту автоматизированную работу МГНОВЕННО, просто выбрать одну команду, нажав клавишу либо щелкнув на панели инструментов.

Это звучит слишком хорошо, чтобы быть правдой, но, используя VBA, можно сделать нечто, называемое МАКРОСОМ. Он в действительности состоит из списка действий, которые надо запомнить для повторного выполнения. Таким образом, макрос не сильно отличается от кулинарного рецепта, состоящего из набора инструкций, в которых говориться, какие действия необходимо выполнить, чтобы что-нибудь приготовить.

Макрос - это набор инструкций, которые сообщают программе (такой как Word или Excel), какие действия следует выполнить, чтобы достичь определенной цели.

Различие, однако, состоит в том, что макрос объединяет все эти инструкции в одном сценарии, который затем можно вызвать с помощью команды меню, кнопки панели инструментов или комбинации клавиш. С этой точки зрения макрос отличается от, скажем, рецепта приготовления хлеба, но похож на автоматическую хлебопекарню, загрузив ингредиенты в которую, можно испечь хлеб одним нажатием кнопки.

Список инструкций, составляющих макрос, как правило, состоит из макрооператоров. Некоторые операторы выполняют особые действия, связанные с выполнением самого макроса, но большинство операторов соответствует командам меню и опциям диалоговых окон приложения, в котором выполняется макрос.

Например, в любом приложении можно закрыть текущее (активное) окно, выбрав команду Закрыть из меню Файл. В макросе VBA следующий оператор позволяет сделать тоже самое: ActiveWindow.Close (АктивноеОкно.Закрыть)

Названия операторов очень информативны, говорят сами за себя о том, что они делают, и не требуется отдельных пояснений.

VBA представляет собой среду программирования, разработанную специально для создания макросов в других приложениях. Возможно, последняя фраза звучит устрашающе, но наибольшим преимуществом языка программированияVBA является то, что он намного проще, чем большинство других языков программирования. Если вы не хотите ничего самостоятельно программировать, то с помощью VBA можете просто записать макрос и назначить его кнопке, расположенной не посредственно в самом документе окна простым перетаскиванием соответствующих элементов управления в окно документа. Другие средства позволяют настраивать меню и панели инструментов. Поэтому в вашем распоряжении есть все необходимое для создания простого сценария, при этом не нужно вручную писать не строчки кода.

Конечно, если вы хотите воспользоваться средствами VBA, вам нужно будет добавить в записанный макрос несколько строк кода самостоятельно. Язык VBA разработан таким образом, чтобы можно было быстро начать программировать на нем, а затем по мере изучения совершенствовать свои навыки. С помощью языка VBA можно почти сразу (без длительного обучения) начать создавать полезные макросы. Данное свойство выгодно отличает VBA от других языком программирования.

Самый простой способ создать макрос - воспользоваться средством записи макросов. Для этого нужно просто запустить средство записи и затем выполнить операции, которые вы хотите автоматизировать (таким образом можно выделить текст, а также выбирать команды меню или изменить опции диалогового окна.) Средство записи переведет все, что вы сделаете, в соответствующие операторы VBA. Результат записи будет сохранен в отдельной области, называемой модулем, откуда можно будет повторить всю процедуру в любое время.

# ÐÐ°ÑÑÐ¸Ð½ÐºÐ¸ Ð¿Ð¾ Ð·Ð°Ð¿ÑÐ¾ÑÑ word Ð¼Ð°ÐºÑÐ¾ÑÑ Ð¾Ð¿Ð¸ÑÐ°Ð½Ð¸Ðµ

Недостатки:

* необходимо хорошее знание персонала для работы в Office
* недостаточная гибкость
* неудобность в работе

# Различные пользовательские программы, разработанные для авто генерации документов.

На примере программа-конструктора С.О.П.О.Р

Программа предназначена для создания и хранения медицинских записей (дневников, первичных осмотров, эпикризов, консультаций) врачами поликлиник и стационаров [удобна в использовании неврологами, кардиологами, терапевтами, хирургами, реаниматологами, стоматологами, мед. психологами, логопедами и другими специалистами]. Присутствуют широкие возможности подстройки программы под нужды конкретного отделения.

Принцип работы программы состоит в следующем: каждому элементу соответствует определенная фраза, при нажатии вносимая в текстовое поле редактора. Набор текста происходит очень быстро при кликах на кнопки и раскрывающиеся списки. При этом в любое время доступно ручное редактирование текста, что значительно облегчает создание рутинных медицинских записей. При сохранении программа автоматически указывает в имени файла тип записи и дату, что позволяет систематизировать и вести базу данных больных, хранимую в текстовом виде. 

Недостатки:

* больше предназначена для стационаров
* для использования на амбулаторном этапе необходима длительная настройка,
* невозможность реализации полностью готового документа

# **5.Моделирование предметной области**

**5.1 Теоретические сведения по теме:**

В приложении используются следующие компоненты win forms.

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

С одной стороны, Windows Forms рассматривается как замена более старой и сложной библиотеке MFC, изначально написанной на языке C++. С другой стороны, WF не предлагает парадигму, сравнимую с MVC. Для исправления этой ситуации и реализации данной функциональности в WF существуют сторонние библиотеки. Одной из наиболее используемых подобных библиотек является User Interface Process Application Block, выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания. Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

Внутри .NET Framework, Windows Forms реализуется в рамках пространства имён System.Windows.Forms.

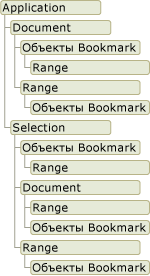
Использованы следующие компоненты Windows.Forms

* Элемент управления CheckBox Windows Forms указывает, включено или отключено какое-либо условие. Обычно он используется для предоставления пользователю выбора типа "Да/Нет" или "Истина/Ложь". Элементы управления типа "флажок" можно объединять в группы для предоставления нескольких вариантов ответа, из которых пользователь может выбрать один или несколько. Это похоже на элемент управления RadioButton, но можно выбрать любое число объединенных в группу элементов CheckBox.
* Элемент управления Windows Forms Button позволяет пользователю щелкнуть его для выполнения действия. На элементе управления Button могут отображаться текст и изображения. При щелчке кнопки мышью элемент управления выглядит так, как будто его нажимают и отпускают.
* Элемент управления CheckBox Windows Forms указывает, включено или отключено какое-либо условие. Обычно он используется для предоставления пользователю выбора типа "Да/Нет" или "Истина/Ложь". Элементы управления типа "флажок" можно объединять в группы для предоставления нескольких вариантов ответа, из которых пользователь может выбрать один или несколько. Это похоже на элемент управления RadioButton, но можно выбрать любое число объединенных в группу элементов CheckBox.
* Windows Forms ComboBox элемент управления используется для отображения данных в раскрывающемся поле со списком. По умолчанию ComboBox элемент управления отображается в виде двух частей: верхняя часть представляет собой текстовое поле, которое пользователь может ввести элемент списка. Вторая часть — полем со списком, отображающий список элементов, из которых пользователь может выбрать его.
* Windows Forms DateTimePicker управления позволяет пользователю выбрать один элемент из списка дат или времени. При использовании для представления даты, он отображается в виде двух частей: раскрывающегося списка с датой, представленных в текст и сетки, которая появляется, если щелкнуть стрелку вниз рядом со списком.
* Windows Forms FileDialog класс является общий базовый класс для OpenFileDialog и SaveFileDialog компонентов. Можно внести изменения в FileDialog класс, который влияет на внешний вид и поведение этих диалоговых окон, в зависимости от версии Windows.
* Windows Forms Label элементы управления используются для отображения текста или изображений, которые не может быть изменено пользователем. Они используются для идентификации объектов в форме, чтобы дать описание какие определенный элемент управления будет делать, если нажата, например или для отображения сведений в ответ на событие во время выполнения или процесс в приложении. Так как Label элемент управления не может получать фокус, он может также использоваться для создания ключей доступа для других элементов управления.
* MaskedTextBox. Описание способа изменения формата данных, когда данные в базе данных не соответствует формату, ожидаемому определения маски.
* Элемент управления Panel. Предоставляет возможность идентифицируемой группировки других элементов управления, а также прокрутку.
* Компонент PrintDialog. Служит для выбора принтера, печатаемых страниц и других параметров печати.
* Элемент управления RadioButton. Представляет набор из двух или более взаимоисключающих вариантов выбора для пользователя.
* Элемент управления TextBox. Позволяет пользователю вводить изменяемый текст из нескольких строк.
* Элемент управления WebBrowser. Содержит веб-страницы и обеспечивает возможности просмотра интернет-страниц в приложении.

Ввод необходимой информации происходит с помощью приведенных контролов, генерация готового документа происходит путем формирования html документа и вывода его в WebBrowser. Формирование html строк занимается класс genHtmText.

Также готовый документ можно открывать в Microsoft Word, используя технологию DOM

Word предоставляет сотни различных объектов, с которыми можно взаимодействовать. Они организованы в виде иерархии, которая точно соответствует пользовательскому интерфейсу. В верхней части иерархии находится объект Application . Этот объект представляет текущий экземпляр Word. Объект Application содержит объекты Document, Selection, Bookmarkи Range . Каждый из этих объектов содержит множество методов и свойств, к которым можно обращаться для работы с объектом и взаимодействия с ним.



На первый взгляд объекты перекрываются друг с другом. Например, объекты Document и Selection являются элементами объекта Application , но объект Document также является элементом объекта Selection . Объекты Document и Selection содержат объекты Bookmark и Range . Перекрытие существует, так как для доступа к объекту одного и того же типа можно использовать различные способы. Например, вы применяете форматирование к объекту Range , но может потребоваться доступ к диапазону текущего выделения, конкретного абзаца, раздела или всего документа.

**5.2** **Обоснование выбора языка программирования**

Прогресс компьютерных технологий определил процесс появления новых разнообразных знаковых систем для записи алгоритмов - языков программирования. Смысл появления такого языка - оснащённый набор вычислительных формул дополнительной информации, превращает данный набор в алгоритм. Язык программирования служит двум связанным между собой целям: он даёт программисту аппарат для задания действий, которые должны быть выполнены, и формирует концепции, которыми пользуется программист, размышляя о том, что делать.

C# (говоря на русском, си Шарп) – это объектно-ориентированное программирование. Он был разработан в 2001 году, инженерами под руководством Андерса Хейлсберга в компании Microsoft. На данное время существует 4 версии языка «си Шарп».

Название «Си Шарп» (от англ. sharp - диез) происходит от музыкальной нотации, где знак диез, прибавляемый к основному обозначению ноты, означает повышение соответствующего этой ноте звука на полутон. Это аналогично названию языка С++, где «++» обозначает, что переменная должна быть увеличена на 1.

К числу принципиально важных решений, которые реализованы корпорацией Microsoft в языке программирования C#, можно отнести следующие: - компонентно-ориентированный подход к программированию (который характерен и для идеологии Microsoft .NET в целом); - свойства как средство инкапсуляции данных (характерно также в целом для ООП); - обработка событий (имеются расширения, в том числе в части обработки исключений, в частности, оператор try); - унифицированная система типизации (соответствует идеологии Microsoft .NET в целом); - делегаты (delegate – развитие указателя на функцию в языках C и C++); - индексаторы (indexer – операторы индекса для обращения к элементам класса- контейнера); - перегруженные операторы (развитие ООП); - оператор foreach (обработка всех элементов классов-коллекций, аналог Visual Basic); - механизмы boxing и unboxing для преобразования типов; - атрибуты (средство оперирования метаданными в COM-модели); - прямоугольные массивы (набор элементов с доступом по номеру индекса и одинаковым количеством столбцов и строк).

Достоинства си-шарпа по книге Биллига:

* C# создавался параллельно с каркасом Framework .Net и в полной мере учитывает все его возможности - как FCL, так и CLR;
* C# является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
* C# является мощным объектным языком с возможностями наследования и универсализации;
* C# является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от С++ к C#;

Сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства С++ как указатели, адресация, разыменование, адресная арифметика;

Благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;

Мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;

Реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

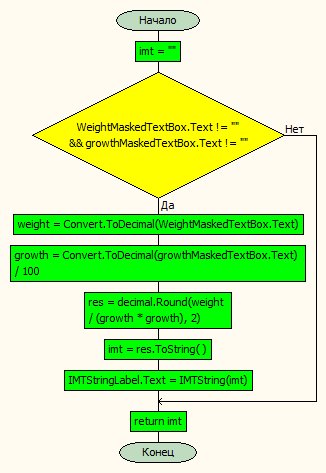
Выделение и объединение лучших идей современных языков программирования делает язык C# не просто суммой их достоинств, а языком программирования нового поколения.

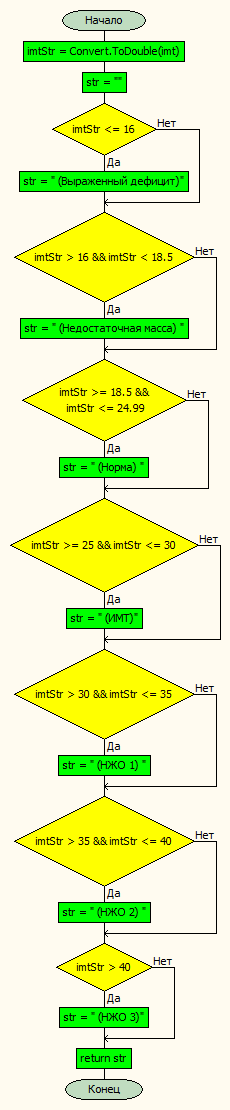
**6.Проектирование программного средства**

**Описание типа данных**

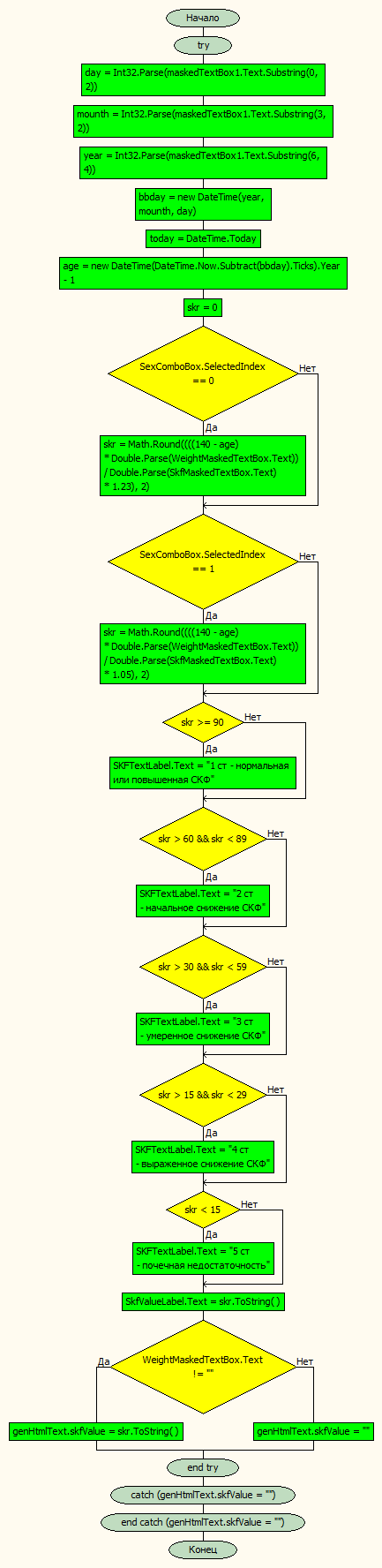
Для ввода информации используются внутренние контроллы в winforms. Элементы для ввода содержаться в классе form1. Данный класс служит для обработки и передачи строк и чисел в класс gentext, который генерирует готовый документ. Каждое изменение конроллов меняет поля класса gentext. Также класс form1 вычисляет некоторые дополнительные вычисления:

* Вычисления ИМТ:

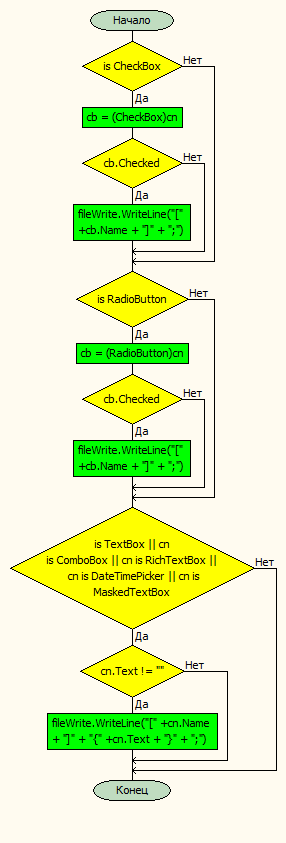




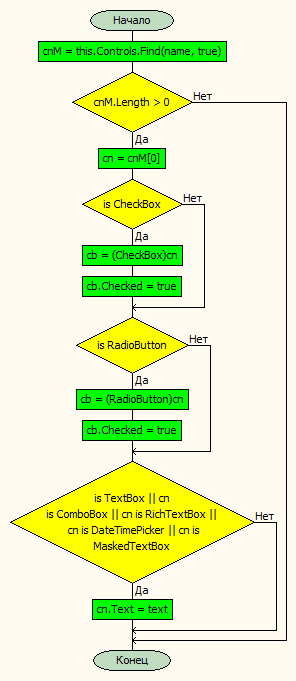
* Вычисления СКФ:



* Сохранение контроллов:

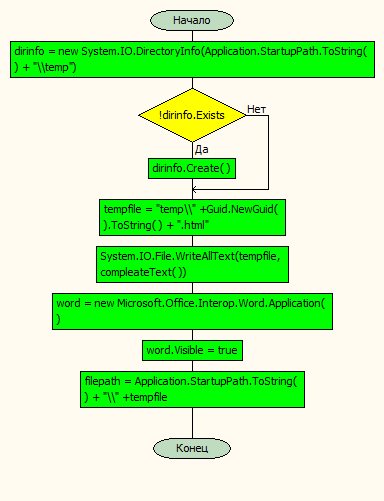


* Загрузка контроллов

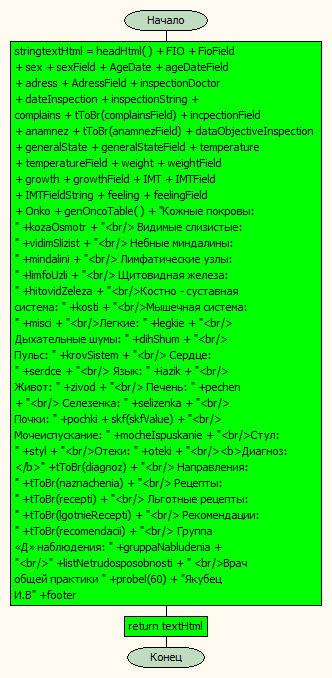


Изменение входных данных генерируют события инициализации класса gentext, в этом классе содержаться следующие методы:

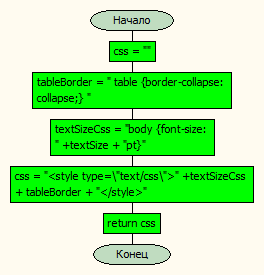
* OpenToWord() открывающий документ Word:



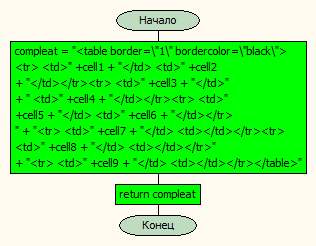
* compleateText() генерирует готовый html документ:



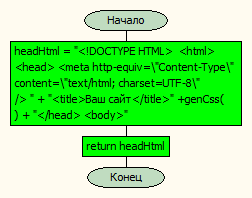
* genCss() задает стили документу:



* genOncoTable() генерирует онко таблицу:



* headHtml() генерирует шапку html:



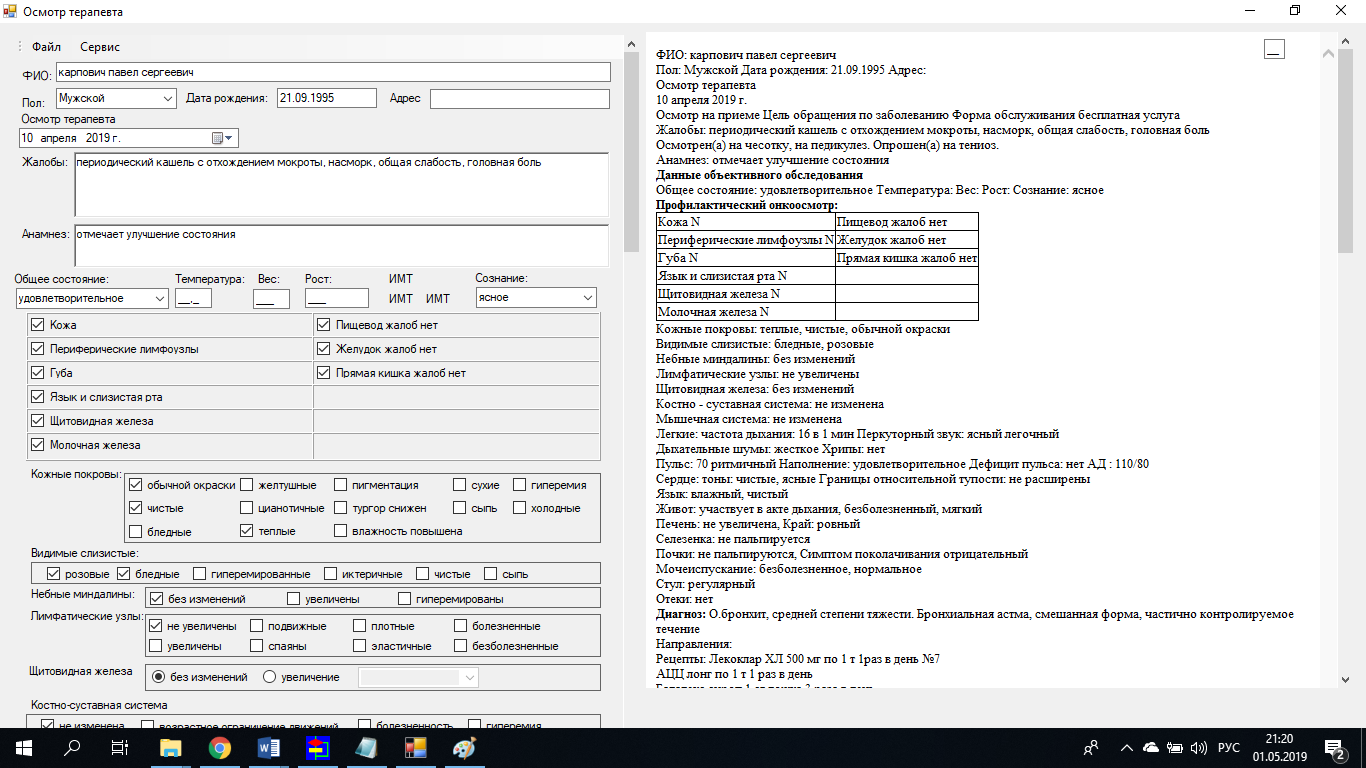
**7.Тестирование ПС**

Тестирование программы - это этап, на котором проверяется, как ведет себя программа на как можно большем количестве входных наборов данных, в том числе и на заведомо неверных.

Основные принципы организации тестирования:

* + 1. необходимой частью каждого теста должно являться описание ожидаемых результатов работы программы, чтобы можно было быстро выяснить наличие или отсутствие ошибки в ней;
    2. следует по возможности избегать тестирования программы ее автором, т.к. кроме уже указанной объективной сложности тестирования для программистов здесь присутствует и тот фактор, что обнаружение недостатков в своей деятельности противоречит человеческой психологии (однако отладка программы эффективнее всего выполняется именно автором программы);
    3. по тем же соображениям организация - разработчик программного обеспечения не должна “единолично” его тестировать (должны существовать организации, специализирующиеся на тестировании программных средств);
    4. должны являться правилом доскональное изучение результатов каждого теста, чтобы не пропустить малозаметную на поверхностный взгляд ошибку в программе;
    5. необходимо тщательно подбирать тест не только для правильных (предусмотренных) входных данных, но и для неправильных (непредусмотренных);
    6. при анализе результатов каждого теста необходимо проверять, не делает ли программа того, что она не должна делать;
    7. следует сохранять использованные тесты (для повышения эффективности повторного тестирования программы после ее модификации или установки у заказчика);
    8. тестирования не должно планироваться исходя из предположения, что в программе не будут обнаружены ошибки (в частности, следует выделять для тестирования достаточные временные и материальные ресурсы);
    9. следует учитывать так называемый “принцип скопления ошибок”: вероятность наличия не обнаруженных ошибок в некоторой части программы прямо пропорциональна числу ошибок, уже обнаруженных в этой части;
    10. следует всегда помнить, что тестирование - творческий процесс, а не относиться к нему как к рутинному занятию.

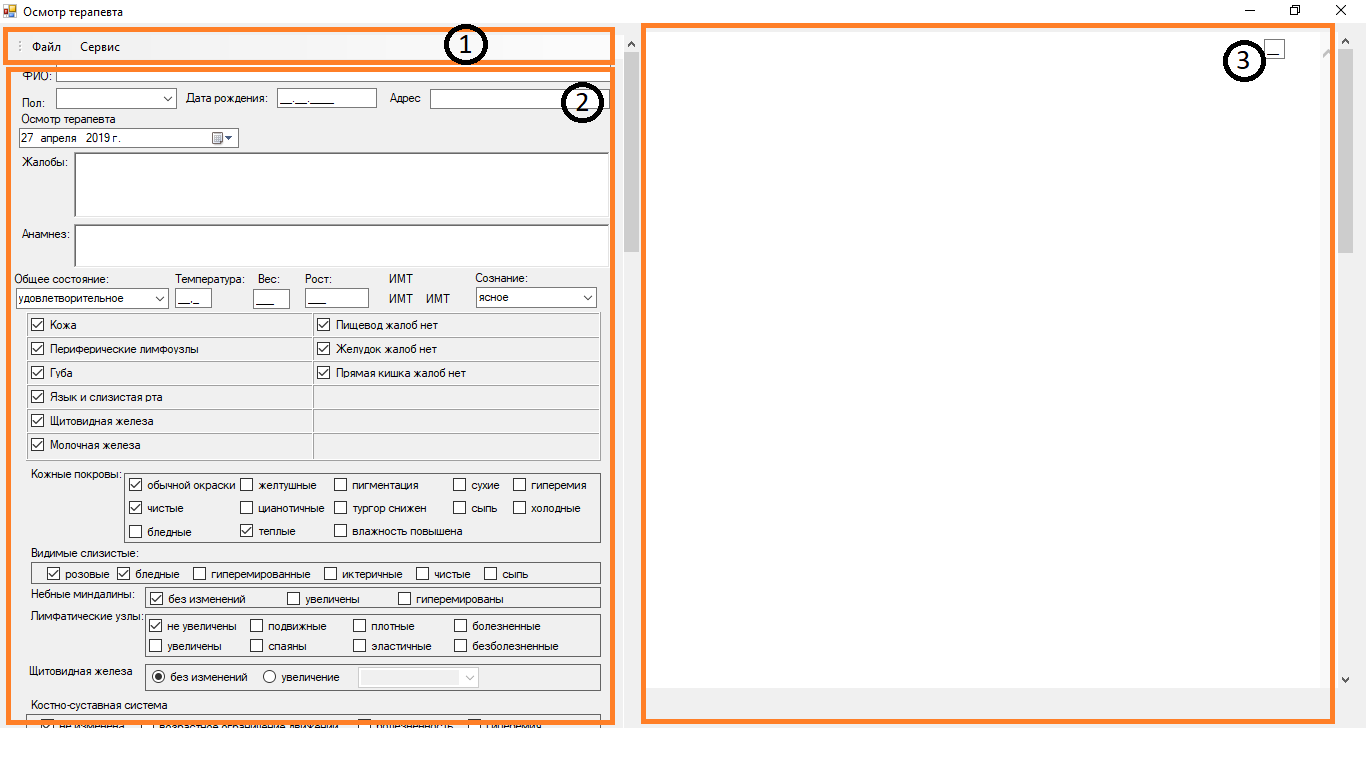
При тестировании программы была выполнена только часть приведенных выше принципов. Тестирование выявило ряд ошибок в алгоритме программы, которые были успешно исправлены, а также были внесены изменения в интерфейс пользователя.



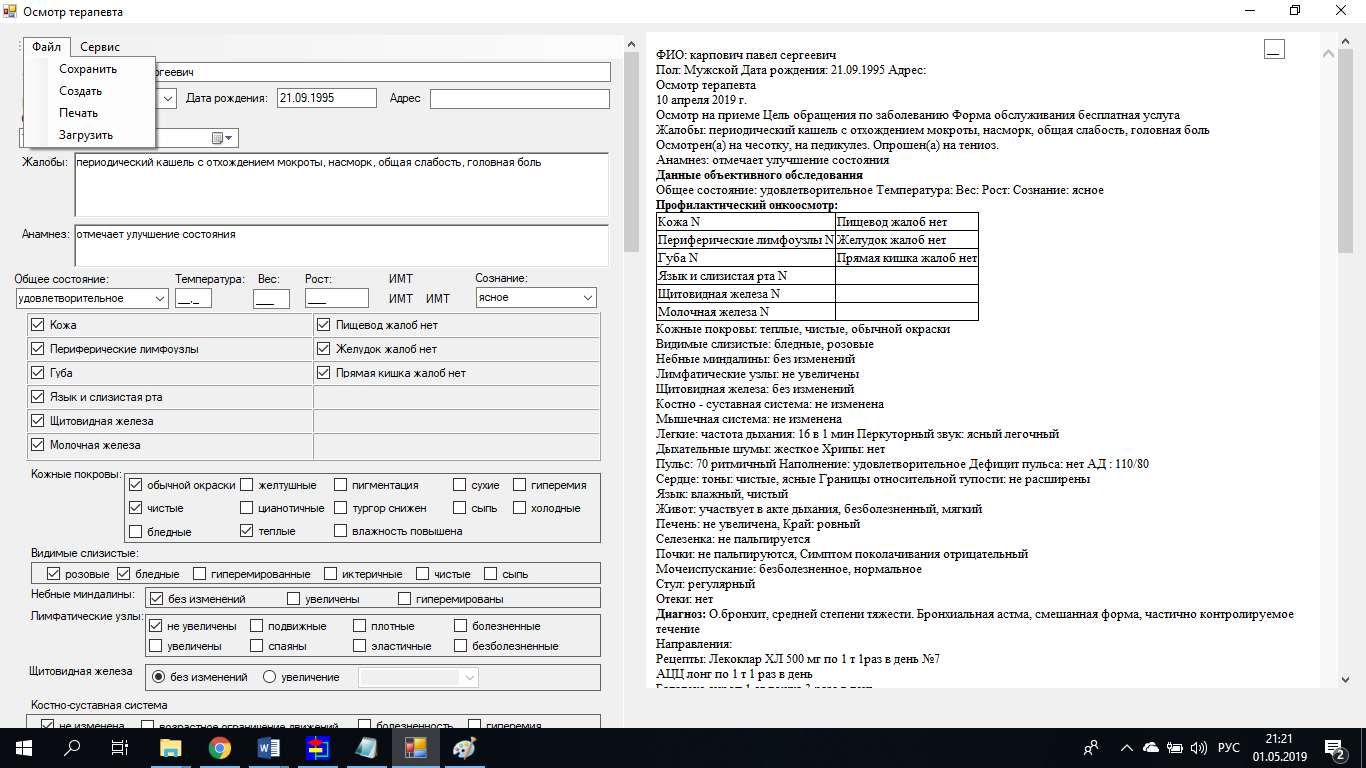
При тестировании программы, были выявлены следующие ошибки:

1. при сохранении документа, сохранялись не все введенные данные
2. некоторые введенные значения могли иметь неверный формат
3. не все элементы управления отображались корректно
4. ошибки при формировании стиля документа
5. неверное преобразование конца строк
6. проблема открытия файлов

**8.Руководство пользователя**



* + - 1. главное меню



Вкладка файл служит для сохранения, создания, печати и загрузки документа.

Вкладка сервис служит для открытия документа в Word.

* + - 1. Панель ввода данных, служит для ввода информации. Стандартные настройки программы содержаться в файле default.
      2. Окно просмотра сгенерированного документа. При вводе информации она автоматически будет выводиться в этом окне.

**9.Заключение**

В данной курсовой работе была выполнена поставленная задача разработать программу для ведения врачом на амбулаторном приеме электронной медицинской карты пациента и сократить временные затраты регистратора за счет автоматизации основных процессов.

**10.Список использованных источников**

1. *Head First C#*, Jennifer Greene, Andrew Stellman (русский перевод: *Изучаем C#*, Д. Грин, Э. Стиллмен).
2. *C# 6.0 and the .NET 4.6 Framework* (7th Edition), Andrew Troelsen, Philip Japikse (русский перевод предыдущего издания: *Язык программирования C# 5.0 и платформа .NET 4.5*, Эндрю Троелсен).
3. *C# 4.0: полное руководство*, Герберт Шилдт.
4. *C# 2010. Ускоренный курс для профессионалов*, Нэш Трей
5. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#, Джеффри Рихтер.
6. C# 7.0 in a Nutshell: The Definitive Reference, Joseph Albahari, Ben Albahari (русский перевод предыдущего издания: C# 6.0. Справочник. Полное описание языка, Джозеф Албахари, Бен Албахари).
7. Essential C# 6.0, Mark Michaelis в сооавторстве с Эриком Липпертом.
8. Effective C# и More Effective C#, Bill Wagner.
9. Programming C# 5.0: Building Windows 8, Web, and Desktop Applications for the .NET 4.5 Framework, Ian Griffiths (русский перевод: Программирование на C# 5.0, Иэн Гриффитс).

**11.Листинг программы**

using System;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace terapevt

{

public partial class Form1 : Form

{

gentext genHtmlText = new gentext();

public Form1()

{

InitializeComponent();

refresh.Visible = false;

webBrowser1.ScrollBarsEnabled = false;

if (!isFirstStart.isOpenFile)

{ defaultFileSet(); }

if (isFirstStart.isOpenFile)

{ openFile(); }

// defaultSet();

}

void defaultFileSet()

{

using (StreamReader defFile = new StreamReader(Application.StartupPath.ToString() + "\\default"))

ControlContainsText(defFile);

}//загружаем файл по умолчанию

void openFile()

{

using (System.IO.StreamReader fileRead = new System.IO.StreamReader(isFirstStart.openFile.OpenFile()))

ControlContainsText(fileRead);

}

public void defaultSet() //стандартные установки

{

cripiVlaznieComboBox.Enabled = false;

cripiSuhiecomboBox.Enabled = false;

genPanel.AutoScroll = true;

genHtmlText.feelingField = feelingComboBox.Text;

feelingComboBox.SelectedIndex = 0;

generalStateComboBox.Text = "удовлетворительное";

foreach (CheckBox item in OnkoTableLayoutPanel.Controls.OfType<CheckBox>())

{ item.Checked = true; }

webBrowser1.DocumentText = genHtmlText.compleateText();

webBrowser1.ScrollBarsEnabled = false;

genHtmlText.dateInspection = examinationDateTimePicker.Text;

feelingComboBox.SelectedIndex = 0;

Koza1checkBox.Checked = true;

Koza2checkBox.Checked = true;

kozac6checkBox.Checked = true;

slizistaya1CheckBox.Checked = true;

slizistaya2checkBox.Checked = true;

mindalini1checkBox.Checked = true;

Limfo1checkBox.Checked = true;

hitovid2RadioButton1.Checked = true;

Kosti1checkBox.Checked = true;

misci1CheckBox.Checked = true;

legkieChastotaMaskedTextBox.Text = 16.ToString();

legkie1CheckBox.Checked = true;

crip1CheckBox.Checked = true;

cripi1radioButton.Checked = true;

pulsMaskedTextBox.Text = 70.ToString();

puls2RadioButton.Checked = true;

napolnenie2RadioButton.Checked = true;

deficit2RadioButton.Checked = true;

serdce1CheckBox.Checked = true;

serdce2CheckBox.Checked = true;

serdceGranica1RadioButton.Checked = true;

Yazik1СheckBox.Checked = true;

Yazik2СheckBox.Checked = true;

Jivot1CheckBox.Checked = true;

Jivot2CheckBox.Checked = true;

Jivot3CheckBox.Checked = true;

Pechen1CheckBox.Checked = true;

PechenKrai1CheckBox.Checked = true;

Selezen1CheckBox.Checked = true;

Pochki1CheckBox.Checked = true;

Pocolachovan1RadioButton.Checked = true;

Moche1СheckBox.Checked = true;

Moche2СheckBox.Checked = true;

Stul1СheckBox.Checked = true;

Otek1СheckBox.Checked = true;

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

private void FioTextBox\_TextChanged(object sender, EventArgs e)

{

genHtmlText.FioField = FioTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}//фамилия

private void SexComboBox\_TextChanged(object sender, EventArgs e)//пол

{

genHtmlText.sexField = SexComboBox.Text;

if (SexComboBox.SelectedIndex == 0)

{

MZelezaCheckBox.Enabled = false;

genHtmlText.cell9 = "";

}

else { MZelezaCheckBox.Enabled = true; genHtmlText.cell9 = "Молочная железа N"; }

SKFChanged(this, e);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void maskedTextBox1\_TextChanged(object sender, EventArgs e)//возраст

{

genHtmlText.ageDateField = maskedTextBox1.Text;

SKFChanged(this, e);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void adressTextBox\_TextChanged(object sender, EventArgs e)//адрес

{

genHtmlText.AdressField = adressTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void examinationDateTimePicker\_ValueChanged(object sender, EventArgs e)//дата приема

{

genHtmlText.dateInspection = examinationDateTimePicker.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void ComplainTextBox\_TextChanged(object sender, EventArgs e)//жалобы

{

genHtmlText.complainsField = ComplainTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void AnamneztextBox\_TextChanged(object sender, EventArgs e)//анамнез

{

genHtmlText.anamnezField = AnamneztextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void generalStateComboBox\_TextChanged(object sender, EventArgs e)//общее состояние

{

genHtmlText.generalStateField = generalStateComboBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void TemperatureMaskedTextBox\_TextChanged(object sender, EventArgs e)

{

genHtmlText.temperatureField = TemperatureMaskedTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}//температура

private void WeightMaskedTextBox\_TextChanged(object sender, EventArgs e)//вес

{

genHtmlText.weightField = WeightMaskedTextBox.Text;

genHtmlText.IMTField = IMTGen();

IMTLabel.Text = IMTGen();

SKFChanged(this, e);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void growthMaskedTextBox\_TextChanged(object sender, EventArgs e)//рост

{

genHtmlText.growthField = growthMaskedTextBox.Text;

genHtmlText.IMTField = IMTGen();

IMTLabel.Text = IMTGen();

webBrowser1.DocumentText = genHtmlText.compleateText();

}

public string IMTGen()//вычисление ИМТ

{

string imt = "";

if (WeightMaskedTextBox.Text != "" && growthMaskedTextBox.Text != "")

{

decimal weight = Convert.ToDecimal(WeightMaskedTextBox.Text);

decimal growth = Convert.ToDecimal(growthMaskedTextBox.Text) / 100;

decimal res = decimal.Round(weight / (growth \* growth), 2);

imt = res.ToString();

IMTStringLabel.Text = IMTString(imt);

}

return imt;

}

string IMTString(string imt)//расшифровка ИМТ

{

double imtStr = Convert.ToDouble(imt);

string str = "";

if (imtStr <= 16) { str = " (Выраженный дефицит)"; }

if (imtStr > 16 && imtStr < 18.5) { str = " (Недостаточная масса) "; }

if (imtStr >= 18.5 && imtStr <= 24.99) { str = " (Норма) "; }

if (imtStr >= 25 && imtStr <= 30) { str = " (ИМТ)"; }

if (imtStr > 30 && imtStr <= 35) { str = " (НЖО 1) "; }

if (imtStr > 35 && imtStr <= 40) { str = " (НЖО 2) "; }

if (imtStr > 40) { str = " (НЖО 3)"; }

return str;

}

private void IMTValueLabel\_TextChanged(object sender, EventArgs e)

{

genHtmlText.IMTField = IMTValueLabel.Text + " " + IMTLabel.Text; ;

webBrowser1.DocumentText = genHtmlText.compleateText();

}//числовое значение ИМТ

private void IMTStringLabel\_TextChanged(object sender, EventArgs e)

{

genHtmlText.IMTFieldString = IMTStringLabel.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}//строковое значение ИМТ

private void feelingComboBox\_TextChanged(object sender, EventArgs e)//сознание

{

genHtmlText.feelingField = feelingComboBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void OncoKozaCheckBox\_Click\_1(object sender, EventArgs e)//онко осмотр кожа

{

if (OncoKozaCheckBox.Checked)

{ genHtmlText.cell1 = "Кожа N"; }

else { genHtmlText.cell1 = "Кожа"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void PichevodCheckBox\_Click(object sender, EventArgs e)//онко осмотр пищевод

{

if (PichevodCheckBox.Checked)

{ genHtmlText.cell2 = "Пищевод жалоб нет"; }

else { genHtmlText.cell2 = "Пищевод жалобы есть"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void OnkoPLCheckBox\_Click(object sender, EventArgs e)//онко осмотр периферические Л/У

{

if (OnkoPLCheckBox.Checked)

{ genHtmlText.cell3 = "Периферические лимфоузлы N"; }

else { genHtmlText.cell3 = "Периферические лимфоузлы"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void zeludokCheckBox\_Click(object sender, EventArgs e)//онко осмотр желудок

{

if (zeludokCheckBox.Checked)

{ genHtmlText.cell4 = "Желудок жалоб нет"; }

else { genHtmlText.cell4 = "Желудок жалобы есть"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void GubaCheckBox\_Click(object sender, EventArgs e)//онко осмотр губы

{

if (GubaCheckBox.Checked)

{ genHtmlText.cell5 = "Губа N"; }

else { genHtmlText.cell5 = "Губа"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void PrKishkaCheckBox\_Click(object sender, EventArgs e)//онко осмотр прямая кишка

{

if (PrKishkaCheckBox.Checked)

{ genHtmlText.cell6 = "Прямая кишка жалоб нет"; }

else { genHtmlText.cell6 = "Прямая кишка жалобы есть"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void RotCheckBox\_Click(object sender, EventArgs e)// онко осмотр язык и слизистая рта

{

if (RotCheckBox.Checked)

{ genHtmlText.cell7 = "Язык и слизистая рта N"; }

else { genHtmlText.cell7 = "Язык и слизистая рта"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void HZelezaCheckBox\_Click(object sender, EventArgs e)//онко осмотр щитовидная железа

{

if (HZelezaCheckBox.Checked)

{ genHtmlText.cell8 = "Щитовидная железа N"; }

else { genHtmlText.cell8 = "Щитовидная железа"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void MZelezaCheckBox\_Click(object sender, EventArgs e)//онко осмотр молочная железа

{

if (MZelezaCheckBox.Checked)

{ genHtmlText.cell9 = "Молочная железа N"; }

else { genHtmlText.cell9 = "Молочная железа"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void Koza1checkBox\_CheckedChanged(object sender, EventArgs e)//панель осмотра кожи

{

genHtmlText.kozaOsmotr = "";

foreach (CheckBox tb in panel3.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.kozaOsmotr += tb.Text;

genHtmlText.kozaOsmotr += ", ";

}

}

genHtmlText.kozaOsmotr = cramblDel(genHtmlText.kozaOsmotr);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void slizistaya1CheckBox\_CheckedChanged(object sender, EventArgs e)//панель осмотра видимых слизистых

{

genHtmlText.vidimSlizist = "";

foreach (CheckBox tb in panel4.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.vidimSlizist += tb.Text;

genHtmlText.vidimSlizist += ", ";

}

}

genHtmlText.vidimSlizist = cramblDel(genHtmlText.vidimSlizist);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void mindalini1checkBox\_CheckedChanged(object sender, EventArgs e)//панель осмотра небных миндалин

{

genHtmlText.mindalini = "";

foreach (CheckBox tb in mindaliniPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.mindalini += tb.Text;

genHtmlText.mindalini += ", ";

}

}

genHtmlText.mindalini = cramblDel(genHtmlText.mindalini);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void Limfo1checkBox\_CheckedChanged(object sender, EventArgs e)//панель осмотра л/у

{

genHtmlText.limfoUzli = "";

foreach (CheckBox tb in limfoUzliPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.limfoUzli += tb.Text;

genHtmlText.limfoUzli += ", ";

}

}

genHtmlText.limfoUzli = cramblDel(genHtmlText.limfoUzli);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void hitovidRadioButton\_CheckedChanged(object sender, EventArgs e)//панель осмотра щитовидной железы

{

if (hitovidRadioButton.Checked)

{

hitovidComboBox.Enabled = true;

genHtmlText.hitovidZeleza = "увеличение " + hitovidComboBox.Text;

}

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void hitovid2RadioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (hitovid2RadioButton1.Checked)

hitovidComboBox.Enabled = false;

{ genHtmlText.hitovidZeleza = "без изменений"; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}//если ЩЖ увеличена, то до какой степени

string cramblDel(string strCrmb)//удаляем лишнюю запятую, если конец предложения

{

if (strCrmb != "")

{ strCrmb = strCrmb.Remove(strCrmb.Length - 2, 2); }

return strCrmb;

}

private void Kosti1checkBox\_CheckedChanged(object sender, EventArgs e)//панель осмотра костной системы

{

genHtmlText.kosti = "";

foreach (CheckBox tb in kostiPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.kosti += tb.Text;

genHtmlText.kosti += ", ";

}

}

genHtmlText.kosti = cramblDel(genHtmlText.kosti);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void misci1CheckBox\_CheckedChanged(object sender, EventArgs e)//панель осмотра мышечной системы

{

genHtmlText.misci = "";

foreach (CheckBox tb in misciPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.misci += tb.Text;

genHtmlText.misci += ", ";

}

}

genHtmlText.misci = cramblDel(genHtmlText.misci);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void legkie1CheckBox\_CheckedChanged(object sender, EventArgs e)//панель осмотра легких

{

genHtmlText.legkie = "";

string legkieChastota = legkieHastotaLabel.Text + " " + legkieChastotaMaskedTextBox.Text + " в 1 мин ";

foreach (CheckBox tb in perkytZvukPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.legkie += tb.Text;

genHtmlText.legkie += ", ";

}

}

if (genHtmlText.legkie != "")

{ genHtmlText.legkie = legkiePerkytlabel.Text + " " + genHtmlText.legkie; }

genHtmlText.legkie = cramblDel(genHtmlText.legkie);

genHtmlText.legkie = legkieChastota + genHtmlText.legkie;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void crip1CheckBox\_CheckedChanged(object sender, EventArgs e)//панель осмотра хрипы

{

genHtmlText.dihShum = "";

foreach (CheckBox tb in shumPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.dihShum += tb.Text;

genHtmlText.dihShum += ", ";

}

}

genHtmlText.dihShum = cramblDel(genHtmlText.dihShum);

genHtmlText.dihShum += " Хрипы: ";

if (cripi2radioButton.Checked)

{

cripiVlaznieComboBox.Enabled = false;

cripiSuhiecomboBox.Enabled = true;

genHtmlText.dihShum += cripi2radioButton.Text + " " + cripiSuhiecomboBox.Text;

}

if (cripi1radioButton.Checked)

{

cripiVlaznieComboBox.Enabled = false;

cripiSuhiecomboBox.Enabled = false;

genHtmlText.dihShum += cripi1radioButton.Text;

}

if (cripi3radioButton.Checked)

{

cripiVlaznieComboBox.Enabled = true;

cripiSuhiecomboBox.Enabled = false;

genHtmlText.dihShum += cripi3radioButton.Text + " " + cripiVlaznieComboBox.Text;

}

if (cripi4radioButton.Checked)

{

cripiVlaznieComboBox.Enabled = false;

cripiSuhiecomboBox.Enabled = false;

genHtmlText.dihShum += cripi4radioButton.Text;

}

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void pulsMaskedTextBox\_TextChanged(object sender, EventArgs e)//панель осмотра кровеносная система

{

genHtmlText.krovSistem = pulsMaskedTextBox.Text + " ";

if (puls1RadioButton.Checked)

{ genHtmlText.krovSistem += puls1RadioButton.Text + " "; }

if (puls2RadioButton.Checked)

{ genHtmlText.krovSistem += puls2RadioButton.Text + " "; }

genHtmlText.krovSistem += napolnenieLabel.Text + " ";

if (napolnenie1RadioButton.Checked)

{ genHtmlText.krovSistem += napolnenie1RadioButton.Text + " "; }

if (napolnenie2RadioButton.Checked)

{ genHtmlText.krovSistem += napolnenie2RadioButton.Text + " "; }

genHtmlText.krovSistem += deficitLabel.Text + " ";

if (deficit2RadioButton.Checked)

{ genHtmlText.krovSistem += deficit2RadioButton.Text + " "; }

if (deficitRadioButton.Checked)

{ genHtmlText.krovSistem += deficitRadioButton.Text + " "; }

genHtmlText.krovSistem += krovsistDavlenielabel.Text + " " + adMaskedTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void SerdceChanged(object sender, EventArgs e)//паель осмотра сердце

{

genHtmlText.serdce = serdceToniLabel.Text + " ";

foreach (CheckBox tb in serdcePanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.serdce += tb.Text;

genHtmlText.serdce += ", ";

}

}

genHtmlText.serdce = cramblDel(genHtmlText.serdce) + " ";

genHtmlText.serdce += serdceGranicaLabel.Text + " ";

if (serdceGranica1RadioButton.Checked)

{ genHtmlText.serdce += serdceGranica1RadioButton.Text; }

if (serdceGranica2RadioButton.Checked)

{ genHtmlText.serdce += serdceGranica2RadioButton.Text; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void iazikChanged(object sender, EventArgs e)//пань осмотря языка

{

genHtmlText.iazik = "";

foreach (CheckBox tb in YazikPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.iazik += tb.Text;

genHtmlText.iazik += ", ";

if (tb.Text == Yazik4СcheckBox.Text)

{ genHtmlText.iazik = cramblDel(genHtmlText.iazik) + " " + YazikTextBox.Text + " "; }

}

}

genHtmlText.iazik = cramblDel(genHtmlText.iazik);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void ZivodChanged(object sender, EventArgs e)//панель осмотра живота

{

genHtmlText.zivod = "";

foreach (CheckBox tb in JivotPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.zivod += tb.Text;

genHtmlText.zivod += ", ";

if (tb.Text == Jivot7CheckBox.Text)

{ genHtmlText.zivod = cramblDel(genHtmlText.zivod) + " " + JivotTextBox.Text + " "; }

}

}

genHtmlText.zivod = cramblDel(genHtmlText.zivod);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void pechenChanged(object sender, EventArgs e)//панель осмотра печени

{

genHtmlText.pechen = "";

foreach (CheckBox tb in pehenKraiPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.pechen += tb.Text;

genHtmlText.pechen += ", ";

}

}

if (!Pechen2CheckBox.Checked)

{

PechenKraiPanel.Enabled = true;

genHtmlText.pechen += "Край: ";

foreach (CheckBox tb in PechenKraiPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.pechen += tb.Text;

genHtmlText.pechen += ", ";

if (tb.Text == PechenKrai3CheckBox.Text)

{

genHtmlText.pechen = cramblDel(genHtmlText.pechen) + " " + PechenKraiMaskedTextBox2

.Text + " ";

}

}

}

}

else { PechenKraiPanel.Enabled = false; }

genHtmlText.pechen = cramblDel(genHtmlText.pechen);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void SelezionkaChanged(object sender, EventArgs e)//панель осмотра селезенки

{

genHtmlText.selizenka = "";

foreach (CheckBox tb in SelezenPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.selizenka += tb.Text;

genHtmlText.selizenka += ", ";

if (tb.Text == Selezen2CheckBox.Text)

{ genHtmlText.selizenka = cramblDel(genHtmlText.selizenka) + " " + SelezenMaskedTextBox2.Text + " "; }

}

}

genHtmlText.selizenka = cramblDel(genHtmlText.selizenka);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void VidSystemChanged(object sender, EventArgs e)//панель осмотра выделительной системы

{

genHtmlText.pochki = "";

foreach (CheckBox tb in PockiLanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.pochki += tb.Text;

genHtmlText.pochki += ", ";

}

}

foreach (RadioButton tb in PockiLanel.Controls.OfType<RadioButton>())

{

if (tb.Checked)

{

genHtmlText.pochki += " " + PocolachovanLabel.Text + " ";

genHtmlText.pochki += tb.Text;

}

}

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void MocheispuskanieChanged(object sender, EventArgs e)//панель осмотра мочеиспускания

{

genHtmlText.mocheIspuskanie = "";

foreach (CheckBox tb in MocheispusPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.mocheIspuskanie += tb.Text;

genHtmlText.mocheIspuskanie += ", ";

if (tb.Text == Moche3СheckBox.Text)

{ genHtmlText.mocheIspuskanie = cramblDel(genHtmlText.mocheIspuskanie) + " " + MocheiTextBox.Text + " "; }

}

}

genHtmlText.mocheIspuskanie = cramblDel(genHtmlText.mocheIspuskanie);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void StylChanged(object sender, EventArgs e)//панель осмотра стула

{

genHtmlText.styl = "";

foreach (CheckBox tb in stylPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.styl += tb.Text;

genHtmlText.styl += ", ";

if (tb.Text == Stul4СheckBox.Text)

{ genHtmlText.styl = cramblDel(genHtmlText.styl) + " " + StylTextBox.Text + " "; }

}

}

genHtmlText.styl = cramblDel(genHtmlText.styl);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void OtekiChanged(object sender, EventArgs e)//панель осмотра отеков

{

genHtmlText.oteki = "";

foreach (CheckBox tb in OtekiPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

genHtmlText.oteki += tb.Text;

genHtmlText.oteki += ", ";

}

}

genHtmlText.oteki = cramblDel(genHtmlText.oteki);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void DiagnozTextBox\_TextChanged(object sender, EventArgs e)//диагноз

{

genHtmlText.diagnoz = DiagnozTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void NaznachChanged(object sender, EventArgs e)//назначения

{

genHtmlText.naznachenia = "";

foreach (CheckBox tb in NaznacheniaPanel.Controls.OfType<CheckBox>())

{

if (tb.Checked)

{

if (tb.Text != Naznachenia13СheckBox.Text)

{

genHtmlText.naznachenia += tb.Text;

genHtmlText.naznachenia += ", ";

}

if (tb.Text == Naznachenia13СheckBox.Text)

{ genHtmlText.naznachenia = cramblDel(genHtmlText.naznachenia) + " " + NaznacheniaTextBox.Text + " "; }

}

}

genHtmlText.naznachenia = cramblDel(genHtmlText.naznachenia);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void ReceptRichTextBox\_TextChanged(object sender, EventArgs e)//рецепты

{

genHtmlText.recepti = ReceptRichTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void LgotaRichTextBox\_TextChanged(object sender, EventArgs e)//льготные рецепты

{

genHtmlText.lgotnieRecepti = LgotaRichTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void RecomendRichTextBox\_TextChanged(object sender, EventArgs e)//рекомендации

{

genHtmlText.recomendacii = RecomendRichTextBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void DispanserСomboBox\_SelectedIndexChanged(object sender, EventArgs e)//группа Д осмотра

{

genHtmlText.gruppaNabludenia = DispanserСomboBox.Text;

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void ListNetrudosposobnostiChanged(object sender, EventArgs e)//лист нетрудоспособности

{

genHtmlText.listNetrudosposobnosti = "";

if (ListNetrudosposobnostiCheckBox.Checked)

{

listNetrudoPanel.Enabled = true;

genHtmlText.listNetrudosposobnosti = "<br/>Лист временной нетрудоспособности: ";

genHtmlText.listNetrudosposobnosti = SeriaLabel.Text + " ";

genHtmlText.listNetrudosposobnosti += SeriaTextBox.Text + " ";

genHtmlText.listNetrudosposobnosti += NomerLabel.Text + " " + nomerbolnTextBox.Text + "<br/>";

genHtmlText.listNetrudosposobnosti += SrokLabel.Text + " " + SrokDateTimePicker1.Text + " " + PriemLabel.Text + " " + dateTimePicker1.Text;

genHtmlText.listNetrudosposobnosti += "<br/>" + PovtorLabel.Text + " " + PovtorDateTimePicker2.Text;

genHtmlText.listNetrudosposobnosti += "<br/>" + "Режим: амбулаторный";

if (ActivePoshCheckBox.Checked)

{

ActivDateTimePicker2.Enabled = true;

genHtmlText.listNetrudosposobnosti += "<br/>" + ActivePoshCheckBox.Text + " " + ActivDateTimePicker2.Text;

}

else { ActivDateTimePicker2.Enabled = false; }

}

else { listNetrudoPanel.Enabled = false; }

webBrowser1.DocumentText = genHtmlText.compleateText();

}

private void panel2\_Paint(object sender, PaintEventArgs e)

{

}

private void сохранитьToolStripMenuItem\_Click(object sender, EventArgs e)//событие меню сохранения файла

{

SaveFileDialog saveControlToFile = new SaveFileDialog();

saveControlToFile.AddExtension = true;

saveControlToFile.DefaultExt = "pacient";

if (FioTextBox.Text != "")

{

saveControlToFile.FileName = FioTextBox.Text;

}

if (saveControlToFile.ShowDialog() == DialogResult.Cancel)

return;

using (System.IO.StreamWriter fileWrite = new System.IO.StreamWriter(saveControlToFile.OpenFile()))

parseControl(genPanel, fileWrite);

}

private void открытьВWordToolStripMenuItem\_Click(object sender, EventArgs e)//открыть документ в word

{

genHtmlText.OpenToWord();

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)//удаляем временные файлы при выходе

{

System.IO.DirectoryInfo di = new System.IO.DirectoryInfo(Application.StartupPath.ToString() + "\\temp");

foreach (System.IO.FileInfo file in di.GetFiles())

{

try

{ file.Delete(); }

catch { };

}

}

private void создатьToolStripMenuItem\_Click(object sender, EventArgs e)//обнуление формы

{

isFirstStart.isOpenFile = false;

Form1 newform = new Form1();

newform.Show();

this.Dispose(false);

genHtmlText = new gentext();

}

private void печатьToolStripMenuItem\_Click(object sender, EventArgs e)//печать документа

{

webBrowser1.ShowPrintDialog();

}

private void maskedTextBox2\_TextChanged(object sender, EventArgs e)//размер текста

{

if (maskedTextBox2.Text != "" && Convert.ToInt32(maskedTextBox2.Text) != 0)

{

genHtmlText.textSize = Convert.ToInt32(maskedTextBox2.Text);

webBrowser1.DocumentText = genHtmlText.compleateText();

}

}

private void saveControl(Control cn, System.IO.StreamWriter fileWrite)//сохраняем значение контролов

{

if (cn is CheckBox)//если выделяется, сохраняется только название

{

CheckBox cb = (CheckBox)cn;

if (cb.Checked)

{ fileWrite.WriteLine("[" + cb.Name + "]" + ";"); }

}

if (cn is RadioButton)

{

RadioButton cb = (RadioButton)cn;

if (cb.Checked)

{ fileWrite.WriteLine("[" + cb.Name + "]" + ";"); }

}

if (cn is TextBox || cn is ComboBox || cn is RichTextBox || cn is DateTimePicker || cn is MaskedTextBox)//если контрол содержит текст

//то сохраняем его

{ if (cn.Text != "") fileWrite.WriteLine("[" + cn.Name + "]" + "{" + cn.Text + "}" + ";"); }

}

private void parseControl(Panel recursPan, System.IO.StreamWriter fileWrite)//рекурсивный осмотр панелей

{

foreach (Control cn in recursPan.Controls)

{

if (cn is Panel)

{ parseControl((Panel)cn, fileWrite); }

saveControl(cn, fileWrite);

}

}

private void SKFChanged(object sender, EventArgs e)//расчет СКФ

{

try

{

int day = Int32.Parse(maskedTextBox1.Text.Substring(0, 2));

int mounth = Int32.Parse(maskedTextBox1.Text.Substring(3, 2));

int year = Int32.Parse(maskedTextBox1.Text.Substring(6, 4));

var bbday = new DateTime(year, mounth, day);

var today = DateTime.Today;

var age = new DateTime(DateTime.Now.Subtract(bbday).Ticks).Year - 1;

double skr = 0;

if (SexComboBox.SelectedIndex == 0)

skr = Math.Round((((140 - age) \* Double.Parse(WeightMaskedTextBox.Text)) / Double.Parse(SkfMaskedTextBox.Text) \* 1.23), 2);

if (SexComboBox.SelectedIndex == 1)

skr = Math.Round((((140 - age) \* Double.Parse(WeightMaskedTextBox.Text)) / Double.Parse(SkfMaskedTextBox.Text) \* 1.05), 2);

if (skr >= 90) { SKFTextLabel.Text = "1 ст - нормальная или повышенная СКФ"; }

if (skr > 60 && skr < 89) { SKFTextLabel.Text = "2 ст - начальное снижение СКФ"; }

if (skr > 30 && skr < 59) { SKFTextLabel.Text = "3 ст - умеренное снижение СКФ"; }

if (skr > 15 && skr < 29) { SKFTextLabel.Text = "4 ст - выраженное снижение СКФ"; }

if (skr < 15) { SKFTextLabel.Text = "5 ст - почечная недостаточность"; }

SkfValueLabel.Text = skr.ToString();

if (WeightMaskedTextBox.Text != "")

{ genHtmlText.skfValue = skr.ToString(); }

else { genHtmlText.skfValue = ""; }

}

catch { genHtmlText.skfValue = ""; };

}

private void загрузитьToolStripMenuItem\_Click(object sender, EventArgs e)//событие меню загрузить файл

{

OpenFileDialog openFile = new OpenFileDialog();

if (openFile.ShowDialog() == DialogResult.Cancel)

return;

isFirstStart.isOpenFile = true;

isFirstStart.openFile = openFile;

Form1 newform = new Form1();

newform.Show();

this.Dispose(false);

genHtmlText = new gentext();

}

void ControlContainsText(StreamReader fileRead)//метод обработки строк сохраненного файла

{

while (!fileRead.EndOfStream)

{

string readLine = fileRead.ReadLine() + "\n";

while (readLine[readLine.Length - 2] != ';')

{ readLine += fileRead.ReadLine() + "\n"; }

if (readLine.LastIndexOf(']') == readLine.Length - 3)

{

readLine = readLine.Replace("[", "");

readLine = readLine.Replace("]", "");

readLine = readLine.Replace(";", "");

readLine = readLine.Replace("\n", "");

openFileControl(readLine);

}

else

{

int nameStart = readLine.IndexOf('[');

int nameEnd = readLine.IndexOf(']');

string name = readLine.Substring(nameStart + 1, nameEnd - nameStart - 1);

int StartText = readLine.IndexOf('{');

int endText = readLine.IndexOf('}');

string text = readLine.Substring(StartText + 1, endText - StartText - 1);

openFileControl(name, text);

}

}

}

void openFileControl(string name, string text = "")//загружаем значение контролов

{

var cnM = this.Controls.Find(name, true);

if (cnM.Length > 0)

{

Control cn = cnM[0];

if (cn is CheckBox)

{

CheckBox cb = (CheckBox)cn;

cb.Checked = true;

}

if (cn is RadioButton)

{

RadioButton cb = (RadioButton)cn;

cb.Checked = true;

}

if (cn is TextBox || cn is ComboBox || cn is RichTextBox || cn is DateTimePicker || cn is MaskedTextBox)

{

cn.Text = text;

}

}

}

private void Form1\_Load(object sender, EventArgs e)

{

ComplainTextBox\_TextChanged(sender, e);

AnamneztextBox\_TextChanged(sender, e);

DiagnozTextBox\_TextChanged(sender, e);

ReceptRichTextBox\_TextChanged(sender, e);

LgotaRichTextBox\_TextChanged(sender, e);

RecomendRichTextBox\_TextChanged(sender, e);

examinationDateTimePicker\_ValueChanged(sender, e);

ListNetrudosposobnostiChanged(sender, e);

}

private void webBrowser1\_DocumentCompleted(object sender, WebBrowserDocumentCompletedEventArgs e)

{

refresh.Checked = true;

}

private void refresh\_CheckedChanged(object sender, EventArgs e)

{

webBrowser1.DocumentText = genHtmlText.compleateText();

}

}

static class isFirstStart

{

public static bool isOpenFile = false;

public static OpenFileDialog openFile;

}

}

using System;

using System.Windows.Forms;

namespace terapevt

{

class gentext//класс генерирует html страницу для контрола webbrouser

{

public int textSize = 10; //поля инициализации документа

string FIO = "ФИО: ";

public string FioField = "";

string sex = "<br>Пол: ";

public string sexField = "";

string AgeDate = " Дата рождения: ";

public string ageDateField = "";

string adress = " Адрес: ";

public string AdressField = "";

string footer = "</body> </html>";

string inspectionDoctor = "<br> Осмотр терапевта<br>";

public string dateInspection = "";

string inspectionString = "<br> Осмотр на приеме Цель обращения по заболеванию Форма " +

"обслуживания бесплатная услуга";

string complains = "<br>Жалобы: ";

public string complainsField = "";

string incpectionField = "<br>Осмотрен(а) на чесотку, на педикулез. Опрошен(а) на тениоз.";

string anamnez = "<br>Анамнез: ";

public string anamnezField = "";

string dataObjectiveInspection = "<br><b>Данные объективного обследования</b>";

string generalState = "<br>Общее состояние: ";

public string generalStateField = "";

string temperature = " Температура: ";

public string temperatureField = "";

string weight = "Вес: ";

public string weightField = "";

string growth = " Рост: ";

public string growthField = "";

string IMT = " ";

public string IMTField = "";

public string IMTFieldString = "";

string feeling = "Сознание: ";

public string feelingField = "";

string Onko = "<br><b>Профилактический онкоосмотр:</b>";

public string cell1 = "";

public string cell2 = "";

public string cell3 = "";

public string cell4 = "";

public string cell5 = "";

public string cell6 = "";

public string cell7 = "";

public string cell8 = "";

public string cell9 = "";

public string kozaOsmotr = "";

public string vidimSlizist = "";

public string mindalini = "";

public string limfoUzli = "";

public string hitovidZeleza = "";

public string kosti = "";

public string misci = "";

public string legkie = "";

public string dihShum = "";

public string krovSistem = "";

public string serdce = "";

public string iazik = "";

public string zivod = "";

public string pechen = "";

public string selizenka = "";

public string pochki = "";

public string mocheIspuskanie = "";

public string styl = "";

public string oteki = "";

public string diagnoz = "";

public string naznachenia = "";

public string recepti = "";

public string lgotnieRecepti = "";

public string recomendacii = "";

public string gruppaNabludenia = "";

public string listNetrudosposobnosti = "";

public string skfValue = "";

public string compleateText() //генерируем полную html строку

{

string textHtml = headHtml() + FIO + FioField + sex + sexField + AgeDate + ageDateField +

adress + AdressField + inspectionDoctor + dateInspection + inspectionString + complains +

tToBr(complainsField) + incpectionField + anamnez + tToBr(anamnezField) + dataObjectiveInspection +

generalState + generalStateField + temperature + temperatureField + weight + weightField + growth

+ growthField + IMT + IMTField + IMTFieldString + feeling + feelingField + Onko + genOncoTable() +

"Кожные покровы: " + kozaOsmotr + "<br/> Видимые слизистые: " + vidimSlizist + "<br/> Небные миндалины: "

+ mindalini + "<br/> Лимфатические узлы: " + limfoUzli + "<br/> Щитовидная железа: " + hitovidZeleza +

"<br/>Костно - суставная система: " + kosti + "<br/>Мышечная система: " + misci + "<br/>Легкие: " +

legkie + "<br/> Дыхательные шумы: " + dihShum + "<br/> Пульс: " + krovSistem + "<br/> Сердце: " +

serdce + "<br/> Язык: " + iazik + "<br/> Живот: " + zivod + "<br/> Печень: " + pechen +

"<br/> Селезенка: " + selizenka + "<br/> Почки: " + pochki + skf(skfValue) + "<br/> Мочеиспускание: " + mocheIspuskanie +

"<br/>Стул: " + styl + "<br/>Отеки: " + oteki + "<br/><b>Диагноз: </b>" + tToBr(diagnoz) + "<br/> Направления: " +

tToBr(naznachenia) + "<br/> Рецепты: " + tToBr(recepti) + "<br/> Льготные рецепты: " + tToBr(lgotnieRecepti) + "<br/> Рекомендации: "

+ tToBr(recomendacii) + "<br/> Группа «Д» наблюдения: " + gruppaNabludenia + "<br/>"

+ listNetrudosposobnosti + " <br/>Врач общей практики " + probel(60) + "" + footer;

return textHtml;

}

public string genOncoTable()//генерация онко таблицы

{

string compleat = "<table border=\"1\" bordercolor=\"black\"> <tr> <td>" + cell1 + "</td> <td>" + cell2 + "</td></tr><tr> <td>" + cell3 + "</td>" +

" <td>" + cell4 + "</td></tr><tr> <td>" + cell5 + "</td> <td>" + cell6 + "</td></tr> " +

"<tr> <td>" + cell7 + "</td> <td></td></tr><tr> <td>" + cell8 + "</td> <td></td></tr>" +

"<tr> <td>" + cell9 + "</td> <td></td></tr></table>";

return compleat;

}

string tToBr(string t)//делаем новую строку html

{

t = t.Replace("\n", "<br/>");

return t;

}

string probel(int count)//генерируем дополнительные пробелы

{

string str = "";

for (int i = 0; i <= count; i++)

{

str += "&nbsp ";

}

return str;

}

public void OpenToWord()//открываем файл word

{

System.IO.DirectoryInfo dirinfo = new System.IO.DirectoryInfo(Application.StartupPath.ToString() + "\\temp");

if (!dirinfo.Exists)

{ dirinfo.Create(); }

string tempfile = "temp\\" + Guid.NewGuid().ToString() + ".html";

System.IO.File.WriteAllText(tempfile, compleateText());

var word = new Microsoft.Office.Interop.Word.Application();

word.Visible = true;

var filepath = Application.StartupPath.ToString() + "\\" + tempfile;

var worddoc = word.Documents.Open(FileName: filepath, ReadOnly: false);

}

string genCss()//генерируем css стиль

{

string css = "";

string tableBorder = " table {border-collapse: collapse;} ";

string textSizeCss = "body {font-size: " + textSize + "pt}";

css = "<style type=\"text/css\">" + textSizeCss + tableBorder + "</style>";

return css;

}

string headHtml()//генерируем шапку html

{

string headHtml = "<!DOCTYPE HTML> <html> <head> <meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\" /> " +

"<title>Ваш сайт</title>" + genCss() + "</head> <body>";

return headHtml;

}

string skf(string skf)//если не высчитано СКФ, то не выводим

{

string skfHtml = "";

if (skf != "" && skf != "0")

skfHtml = "<br/> СКФ: " + skf + " ";

return skfHtml;

}

}

}